NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

AGENT-BASED SIMULATION OF MILITARY OPERATIONS OTHER THAN WAR SMALL UNIT COMBAT

by

Ronald F. A. Woodaman

September 2000

Thesis Advisor:	Arnold H. Buss
Second Reader:	Gordon H. Bradley

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE September 2000	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE
Agent-Based Simulation of Military Operations Other Than War Small Unit Combat

5. FUNDING NUMBERS

6. AUTHOR(S)
Woodaman, Ronald F. A.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

A significant challenge to the Armed Forces today is the development of tactics, techniques, procedures, and equipment that will enable success in the small-scale combats that characterize Military Operations Other Than War (MOOTW). This thesis develops an agent-based simulation methodology for modeling MOOTW combat scenarios. The methodology combines agent-based modeling with discrete event simulation in a software package called AgentKit. AgentKit is used to model a riot control problem for an experiment that pits two kinds of tactics against two different kinds of crowds. This simulation yields insights into the scenario modeled and demonstrates the usefulness of agent-based simulation for the exploration of tactical concepts in a MOOTW context.

14. SUBJECT TERMS
Agents, Modeling and Simulation, Object Oriented Programming, Java, Military Operations Other Than War, Riot Control, Peacekeeping

15. NUMBER OF PAGES 74

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 289)
Prescribed by ANSI Std. 239

i

THIS PAGE INTENTIONALLY LEFT BLANK

AGENT-BASED SIMULATION OF MILITARY OPERATIONS OTHER THAN WAR SMALL UNIT COMBAT

Ronald F. A. Woodaman
Major, United States Marine Corps
BS, U.S. Naval Academy, 1987

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS ANALYSIS

from the

NAVAL POSTGRADUATE SCHOOL
September 2000

Author: _____

Ronald F. A. Woodaman

Approved by: _____

Arnold H. Buss, Thesis Advisor

_____
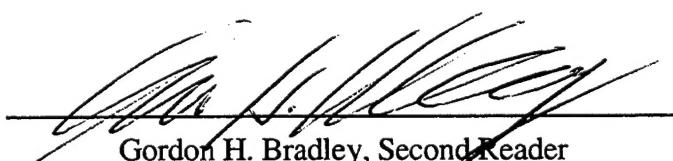
Gordon H. Bradley, Second Reader

_____
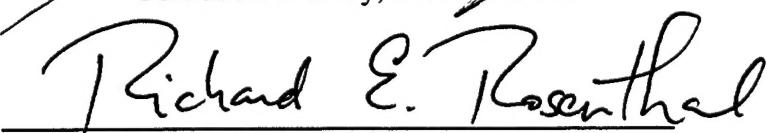
Richard Rosenthal, Chairman
Department of Operations Research

THIS PAGE INTENTIONALY LEFT BLANK

ABSTRACT

A significant challenge to the Armed Forces today is the development of tactics, techniques, procedures, and equipment that will enable success in the small-scale combats that characterize Military Operations Other Than War (MOOTW). This thesis develops an agent-based simulation methodology for modeling MOOTW combat scenarios. The methodology combines agent-based modeling with discrete event simulation in a software package called AgentKit. AgentKit is used to model a riot control problem for an experiment that pits two kinds of tactics against two different kinds of crowds. This simulation yields insights into the scenario modeled and demonstrates the usefulness of agent-based simulation for the exploration of tactical concepts in a MOOTW context.

THIS PAGE INTENTIONALLY LEFT BLANK

DISCLAIMER

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. The reader is cautioned that computer programs developed in this research may not have been exercised for all cases of interest. While every effort has been made, within the time available, to ensure that the programs are free of computational and logic errors, they cannot be considered validated. Any application of these programs without additional verification is at the risk of the user.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGEMENT

The author would like to express his thanks to the many people who were instrumental to the completion of this thesis. To Cpts. Maximo Moore and Robert Bradford, USA, for being great sounding boards. To Dr. Paul Sanchez for providing helpful advice and inspiration throughout the this thesis. To John Hiles who taught me far more about agents than I wanted to learn. To Dr. Gordon Bradley, both for his intellectual insight as well as his terrific support as a Second Reader. To Dr. Arnold Buss, for his vision and expertise in computer modeling. Finally, to my wife Michelle, for without her loving support and patience none of this would have been possible.

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

Security during Military Operations Other Than War (MOOTW) presents special challenges to the U.S. Armed Forces. Of particular concern to this thesis is the encounter between a small unit of peacekeepers and a rioting crowd. This thesis develops a model of this scenario with the goal of exploring tactical concepts. The chosen method of modeling is driven by the peculiarities of MOOTW conflict.

These kinds of small-scale encounters are very much dependent upon the actions of individuals. The success of MOOTW missions in today's era of intense media scrutiny can be easily negated by the failings of individual members of the forces conducting these missions. Often, these failings can be attributed to a lack of adequate preparation, leadership, or simply faulty tactics, techniques, and procedures (TTP). Since the primary goal in peacekeeping is to minimize violence and casualties, both to the force and to the local populace, improved TTP's are an important step. The model we desire should reflect the effect of individual actions, capture the process by which casualties occur, and allow the implementation of different TTP's.

This thesis develops an agent-based simulation methodology in order to model a riot control scenario. The goal of generating the simulation is to gather a better understanding of the process of MOOTW small-scale encounters. Because these are still abstract models, the results of these simulations should not be viewed as rigorous predictions about the relative worth of one tactical concept versus another. In testing different tactical concepts and studying the functioning of the model, the researcher can gain insight as to why a tactic may work in a certain setting. This insight enables the generation of better TTP's.

The simulation was generated with a software package, called AgentKit, written in Java by the author. It is an extension of Simkit, a discrete-event simulation library. The agent model used is a derivation of John Holland's software agent.

The scenario pits a small peacekeeper detachment guarding a valuable site against the attack by an angry mob. The mob transitions from an initially peaceful state to a violent state where they assault the peacekeepers with stones. The peacekeepers are armed with imperfect non-lethal weapons. Their goal is to defend the site until the arrival

of reinforcements, which is fixed for the scenario. Two Measures Of Effectiveness (MOE) are used: expected hits per peacekeeper and expected hits per rioter. It should be noted that achieving a minimum for the second MOE is meaningless if the first MOE is high - it would indicate that the rioters are winning. Two tactics are tested: a reactive tactic that only returns fire, and a proactive tactic that fires on the most violent crowd members to try and break up the crowd. Two forms of mobs are generated: one with an explicit leader and one without.

The simulation was animated with simple caricatures that enabled the user to compare results with observed actions. The animation assisted in determining the causes of certain outcomes.

Experimental results showed that there was a significant drop in casualties for the peacekeepers and the rioters with the use of the proactive tactic. The proactive tactic proved capable of breaking up the crowd most of the time. Leadership in the crowd did not prove to be a significant factor statistically. This may have been a reflection of some of the decisions made in the design of the experiment as opposed to either a) a fault of the model, or b) an indication of the threat posed by crowd leadership.

This thesis demonstrated how this method of simulation may be used to develop insight into the dynamics of a riot control scenario. Furthermore, it highlighted the potential usefulness of agent-based simulation to serve as a tool in the development of tactical concepts.

I. INTRODUCTION

A. AREA OF RESEARCH

Since the end of Cold War, the U.S. and Allied armed forces have been increasingly committed to Military Operations Other Than War (MOOTW). MOOTW places a greater demand on the training and preparedness of individual service members because of the potential for individual actions at the lowest level to have profound impact on the success of these operations. The ability of small units of security forces to succeed against a variety of low-intensity threats dictates the success of MOOTW. Riot control presents a special challenge.

Research by the Armed Forces into MOOTW combat has intensified in recent years. However, while research into the more traditional realms of military force application has benefited from a significant body of simulation models, MOOTW researchers have little to choose from in the way of simulation.

Software agents are a relatively new paradigm in simulation. They provide for the capability to model complex issues from the bottom up. Many complex systems have a seemingly ordered behavior at the aggregate level that is not easily explained given an understanding of the behavior of the individual components that form the system (Holland, 1995). Researchers have employed agent-based simulations to study a variety of phenomena from natural processes such as bee hives and ant colonies to human ones such as cooperative game strategies. Recently, agent-based simulations have been applied to warfare. This thesis develops an agent-based simulation methodology in order to model a riot control scenario. This methodology can serve as a tool for the study of security and tactics for forces involved in Military Operations Other Than War (MOOTW).

B. MILITARY OPERATIONS OTHER THAN WAR

Since 1993 U.S. and Allied forces have increasingly been committed to Military Operations Other than War (MOOTW), particularly operations involving peacekeeping and humanitarian assistance. MOOTW presents a broader array of challenges than conventional warfare. Instead of being able to focus on the destruction of the enemy, leaders at every level are encumbered by the primacy of political considerations in MOOTW (Joint Pub. 3-07, 1995). Often, military forces do not have a readily identifiable foe on which to focus. Instead, the mission is often ambiguous and shifting with numerous parties involved: criminal groups, the general populace, warlords, ethnic minorities, Non-Governmental Organizations (NGO), allied armed forces, and other government agencies. In Bosnia, for instance, the NATO Stabilization Force (SFOR) peacekeepers have to deal with a population comprised of three separate, mutually antagonistic ethnic groups. The peacekeeping effort is further complicated by the diverse political objectives of the nations supporting the peacekeeping process. (Scharfen, 1998)

Within the services, MOOTW is having an increased impact on armed forces training, equipment, and doctrine. In recent years a great deal of effort has gone into preparing for campaign level success in these operations. U.S. forces can be very successful at this level of operations because of their ability to conduct the substantial logistic effort that accompanies these kinds of operations. The U.S. forces also provide a wealth of services: medical, engineering, transportation, administration, policing, legal, and security. (Joint Pub. 3-07, 1995) The many MOOTW deployments in the 1990s have improved U.S. forces' ability to work with not just Allied forces and other government agencies but Non-Government Organizations as well. (Walter, 1998)

The unfortunate part of this rosy picture is that we live in the age of the sound bite and CNN. A MOOTW operation may accomplish all of its other stated objectives - feed the hungry, keep warring parties separated, enforce treaties, rebuild infrastructure - but in the eyes of the world be plagued by scandal due to the failings of individual troops to handle the challenges of MOOTW.

The little incidents of human failure that get swallowed up by the aggregation of violence in conventional warfare rise prominently to the fore in today's era of intense

media scrutiny. The success of the Kosovo operation has been tarnished by a U.S. soldier's rape and murder of a Kosovar child. His unit is now under investigation for possible systematic abuses of the local populace, which may have set the stage for the soldier's heinous crime. Earlier in the Balkans, four U. S. Soldiers surrendered without a fight to Serbian forces that crossed the border into Macedonia, arguable because of the failure of their unit to institute proper security measures. During the Vietnam War, the lack of training and mental preparedness that drove National Guard troops to fire on unarmed college students at Kent State assisted in galvanizing the anti-war movement. Arguably, the ramifications from the failures of individuals in MOOTW have far greater reach than in conventional warfare.

In MOOTW, decision making is forced down to a far lower level than in conventional warfare. In a conventional war, individual fighting men have fewer decisions to make. For instance, the decision to engage is often a matter of simply spotting an enemy within range of organic weapons. In MOOTW, individual security personnel have to make difficult decisions, deciding what is an incident that justifies a violent response and to what level, non-lethal or lethal. Squad leaders have to know when to call for reinforcements against a developing riot situation or when simply to talk to a gathering of locals frustrated with the lack of services. A rifleman manning a check point needs to do more than check identification. He or she has to scan the area constantly for a variety of threats, from snipers in windows to thieves seeking a valuable piece of American-made equipment. In Vietnam, when U.S. troops took fire, their typical response was massive, indiscriminate amounts of fire. Troops in MOOTW must follow Rules of Engagement, which often require a positive target identification of the target and a clear backstop (no bystanders) before permitting return fire. Further, with the employment of non-lethal weapons, there are typically technical restrictions on the use of the weapon to maximize the probability of not killing the target. (Kenny, 2000)

Individuals assigned to security operations in MOOTW are typically armed for war but are ostensibly waging peace. While they must confront violence, destruction of the opponent is an option of last resort, if at all. The potential encounter is characterized by the asymmetries of the participants' combat potential and of their intents. While peacekeeping forces have the desire to minimize conflict, the populace may likely harbor

3

elements who seek to re-ignite conflict. Peacekeeping troops carry all manner of very lethal as well as non-lethal devices. Meanwhile, the populace they face will have very low individual combat potential, but collectively be capable of vast destruction (witness the Los Angeles riots of 1992). Operating from within the safety blanket of the population one may find various criminal groups, terrorists, and guerilla fighters. Thus, the population is often a faceless mass while the individual MOOTW troops are few and very obvious. (Allen, 1995)

The success of MOOTW, as measured by the media, the public, and observers abroad, depends ultimately on individual Soldiers and Marines making the appropriate decisions when confronted with a violent encounter. Arguably, the biggest challenge is developing the appropriate tactics, techniques, and procedures (TTP) in combination with the right equipment that enable the success of troops at the point of potential conflict.

The services in recent years have begun to focus much of their research and development on MOOTW small unit encounters and combat. Much attention throughout the Department of Defense has focused on the development and testing of better non-lethal weapons and associated equipment. The Marine Corps has conducted several Warfighting Experiments to test both tactics and equipment in realistic environments. These Warfighting Experiments and other such field tests, while immensely valuable, are also expensive, difficult to conduct, and require months of intense planning and coordination. The Marine Corps, as well as other Armed Forces, could benefit from the ability to simulate small-scale MOOTW encounters and use these simulations to supplement their ongoing development efforts.

C.   OBJECTIVES AND RESEARCH QUESTIONS

This thesis examines a particular aspect of MOOTW conflict: riot control. It seeks to model a generalized scenario abstracted from a specific civil disorder episode in Guantanamo Bay, Cuba, in 1994. The desire is to be able to generate a model that will support the exploration of tactical concepts for riot control. We will argue that simulation is the best approach with which to start, as it seems to be modeling method that will most likely aid in the examination of different tactics and crowd dynamics. It is

4

possible that insights gained from the simulation may open the door to other modeling methods.

Mission Area Analysis, Requirements Generation, Concept Development, and Doctrine Development are all processes that fall under the general term of Combat Development and fulfill various roles in the overall effort of preparing the force for future threats or how to better face today's threats. Each of these processes could benefit from the ability to simulate and test MOOTW scenarios, whether it be to validate concepts, examine new requirements, model future threat, or develop doctrine. The U. S. Marine Corps conducts combat development at the Marine Corps Combat Development Center (MCCDC) using methods such as computer-based simulation, wargaming, and field-testing. As mentioned above, the Marine Corps' Warfighting Lab conducts extensive field tests of concepts and equipment, much of it oriented towards MOOTW. While MOOTW has been an area of focus for the Marine Corps' combat development effort for much of the past decade, MCCDC has little in the way of computer simulations that specifically address MOOTW combat. Most of the research with computer simulations has been done using large-scale warfare aggregate models to examine campaign level issues such as force deployment, employment, and logistics.

Two questions arise: what should be required of simulations of security operations in MOOTW; and are there current warfare models that can be adapted to this use. Warfare simulations come in two primary varieties: large-scale and small-scale. Large-scale models such as the U.S. Army's Concept Evaluation Model (CEM), used during Desert Storm, aggregate combat (Appleget, 1995). They provide a deterministic prediction of the expected results at a campaign scale. Small-scale models of combat such as JANUS, another U.S. Army simulation system, attempt to model warfare discretely at the entity or small unit level (typically tank or infantry squad) and model battles at the company and battalion level. Clearly, a small-scale model is needed to model security operations because of the impact of individuals on the outcome.

The problem with simulating MOOTW is that the individual participants are not strictly combatants. Entities have choices. For example, in the mid-1990's Cuban refugees were being housed in refugee camps at the U.S. Naval Base at Guantanamo, Cuba. These refugees became unruly over their conditions of the camp and rioted

5

demanding to leave. These refugees consisted of various groups: men, women, children, the elderly, the ill, the disinterested, the criminally minded. The crowd had leaders, agitators, fighters, supporters, and bystanders. The dynamics of the situation were highly charged with violent confrontations developing very quickly. The tactics of the security forces and the crowd changed day to day as they each adapted to the other's previous actions. The factors that went into whether the crowd rioted, merely protested, or failed to form at all were varied and complex (Allen, 1995). The choices these entities make are a function of their internal state. This state can vary widely from one situation to the next. The issue of the internal state of simulation entities is one that most current combat models, simulation or otherwise, fail to capture.

Allen's (1995) article indicates that crowd dynamics can be viewed as a non-linear combination of the individual decisions of a heterogeneous mass of people. For example, one tactic employed by the security forces was the use of snatch teams against crowd agitators. A heavily armed team entered the crowd suddenly and snatched an agitator before the crowd turned. The agitators, as it turned out, had a synergistic effect on the other crowd members and their removal defused the situation. Thus, the effects of individual decisions can have a non-linear affect on the system as a whole.

Ultimately, the problem with current small-scale combat simulations is that they are purely combat-oriented. Entities are partitioned into one of two opposing sides and then allowed to fight in accord with a hard-coded set of rules that represent doctrine, tactics, and capabilities. For MOOTW, the simulation tool should allow for more than two sides, allow the rules governing individual entities to be changed, and give the individual entities the ability to choose their action: to fight, to flee, to abstain, to threaten, etc.

Agent-based simulations may provide an answer. They provide the entity-level representation that seems best suited to the MOOTW problem. They also possess some level of autonomy that is needed to model the issue of choice. As simulations, they grant the level of flexibility necessary to examine different tactical concepts.

D.   SOFTWARE AGENTS

The study of software agents is relatively recent and includes concepts such as intelligent agents, multi-agent systems, bottom-up simulation, robotics, artificial life, and distributed artificial intelligence. Agents have a broad array of uses beyond simulation, from "intelligent" server software to feed back control. This variety of uses has led to a vagueness surrounding the term agent, its function, and its characteristics. The published authoritative sources in the field often disagree with one another.

This thesis relies on two primary sources: Holland (1995) and Ferber (1999). Ferber's work provides a more comprehensive approach to software agents and the computer science issues involved. Holland is more interested in modeling complex adaptive systems (CAS) and sees software agents as the logical tool for their study.

Ferber describes two broad research objectives in this field of study: the study of the self-organizing systems when several autonomous entities interact (what Holland would call a CAS), and the exploitation of this mechanism to create distributed artifacts that can accomplish complex tasks through cooperation and interaction. The latter approach is concerned with a new approach to artificial intelligence. The first approach is what concerns this thesis.

Axelrod (1997) describes agent-based modeling as a third way of doing science in contrast to induction and deduction. Induction can be described as the discovery of patterns in empirical data. Deduction consists of specifying a set of axioms and then proving the consequences of these assumptions. Agent-based modeling, like deduction, starts with a set of explicit assumptions – the rules of the simulation world. Instead of deriving a proof from these deductions, the agent-based simulation generates data that can be analyzed inductively. In contrast to pure induction, where the data come from the real world, the simulated data come from a rigorously specified set of rules of the modeler's creation. Axelrod concludes, " Whereas the purpose of induction is to find patterns in data and that of deduction is to find consequences of assumptions, the purpose of agent-based modeling is to aid intuition." (Axelrod, 1997, pg. 4) This implies that our expectation from agent-based modeling should not be to obtain predictions about a complex system's future state but rather to gain insight into the system.

7

As alluded to earlier, finding a concise definition of the term agent as used in computer science is much like the search for a rigorous definition of intelligence: no consensus exits. Holland avoids a definition, but goes on to describe agents as the models used to represent individual decision making entities that interact in the physical world, be they people or organizations. Ferber attempts to find a minimal common definition with the following:

An agent is a physical or virtual entity which:
   a) is capable of acting in an environment,
   b) can communicate directly with other agents,
   c) is driven by a set of tendencies (in the form of individual objectives or a
   satisfaction/survival function which it tries to optimize),
   d) possesses resources of its own,
   e) is capable of perceiving its environment (but to a limited extent),
   f) has only a partial representation of its environment (and perhaps none at all),
   g) which possesses skills and can offer services,
   h) which may be able to reproduce itself,
   i) whose behavior tends towards satisfying its objectives, taking account of the resources
   and skills available to it and depending on its perception, its representations and the
   communications it receives. (Ferber, 1999, pg. 9)

He does concede that extreme examples exist that do not strictly meet some of these categories. Purely communicating agents have no real environment to perceive. Their "environment" may be the server on which they monitor messages from other nodes on a network. Tropistic agents have no internal reasoning ability and are driven purely by their environment. They can be likened to an ecoli bacterium swimming up a sucrose gradient. These two kinds of agents represent extremes and each fall into what Ferber describes as the two schools of thought on multi-agents systems. The 'cognitive' school espouses systems of small numbers of 'intelligent' agents. The agents possess large knowledge bases and communication skills. They interact, cooperate, and resolve conflict among themselves to develop solutions to complex issues. These kinds of agents are used more often in the study of the sociology of organizations and of small groups.

8

Cognitive agents are described as possessing a symbolic representation of the world from which they can reason. For the purposes of computer modeling, the user would have to use specialized computer languages that support the required symbolic knowledge representation, such as Prolog (Andrade, 2000).

The 'reactive' school does not believe that agents themselves must be intelligent for the system itself to demonstrate intelligent behavior overall. Reactive agents reason at a sub-symbolic level and can be modeled on a computer with more general-purpose languages. They are simple by design and easier to model. This facilitates simulating large numbers of agents, a vital capability to effectively simulate complex, multi-entity systems. Holland (1998) provides an example of an agent-based simulation of an anthill that showed how the interactions of simple entities can lead to complex behavior. Holland claims these kinds of multi-agent systems can be used to model complex adaptive systems; that is, aggregations of many individual entities whose simple individual interactions yield complex behavior. Holland's focus is to devise a way to study in a computer-generated laboratory the concepts of emergence, complexity and adaptivity.

Adaptation is another term that is frequently used in the literature, but whose meaning is sometimes vague. For the purposes of this thesis, let us view two levels of adaptation. The first is the adaptation of the system to change. An aggregation of reactive agents will adapt to new inputs from its environment, but internally the individual entities in the system remain unchanged. Taking Holland's example of the anthill, we can envision the anthill when it is disturbed by a passerby's shoe. The ants appear chaotic but order rapidly emerges as the ants settle back into a variety of tasks, from rebuilding the nest to gathering food. It all appears very ordered but it is known that there is no central executive. The appearance of order appears from the ants interacting and following their simple rules of behavior.

A higher order version of adaptation is internal change. Learning and biological evolution are examples of this kind of adaptation. Learning, for a reactive agent, can be interpreted as the generation of better-adapted rule sets. Biological evolution can be viewed as adaptation of the gene pool to the environment. This kind of agent adaptation could be generated via computational evolution, as described separately by Holland

(1995) and Fogel (2000). This method would seek to employ evolutionary computational methods, to include genetic algorithms, to generate "more fit" agents by evolving new rule sets. Essentially, the problem has changed from modeling a set of internally immutable agents to one where the agents are allowed to evolve in an attempt to optimize some internal objective function. Examples include the probability of reproduction for a biological agent, or the probability of survival for a virtual soldier. A method that offers the possibility of being able to evolve new rules (example: better tactics), is tremendously attractive but offers daunting challenges. A more feasible method to evolve new rules would be to use the insight gained from iterations of the model to evolve better the rules.

It is clear, then, that for the purposes of this thesis, the reactive agent type is most suitable for modeling individuals in small-scale MOOTW scenarios. Since there are no simple off-the-shelf implementations of agents suitable for this kind of simulation, the author needed to find a suitable model for implementation. Holland's agent (1995), to be discussed in the following chapter, was chosen because of his clear and detailed description of its structure and function.

E.   AGENTS IN WARFARE SIMULATION

In recent years, agent-based simulation has been applied to the study of warfare. Ilyanchinski (1997) created a software simulation called Irreducible Semi-Autonomous Agent Combat (ISAAC), using agents to model warfare as a complex adaptive system. Emergence, complexity, and complex adaptive systems (CAS) are all topics of study under the relatively new field called the New Sciences. Researchers in these fields seek scientific principles believed to rule the behavior of complex systems and their ability to adapt to change; e.g., economies, ecologies, immune systems. Ilyachinski (1996) has argued that warfare can possibly be viewed as a CAS given that war is made up of agents interacting at numerous levels and, despite the chaos, order can be seen.

There is another important argument for exploring the use of agent models as a warfare-modeling tool. Most warfare models have at least one of two shortfalls: the assumptions of rational decision making and determinism. Analytical models depend heavily on the premise of rational decision making - that role players in an analytical

problem all have complete knowledge and will always make the optimum choice. These assumptions are often vital to making the problem analytically tractable. But humans often make less than completely rational decisions and they almost never have complete knowledge. This is reflected in what Clausewitz (1940) called the fog of war.

Many warfare simulations aggregate combat using attrition rates. The results generated under these simulations are essentially the long-term average results of that specific encounter – the expected results. The history of warfare, though, is rife with examples of battles that had very unexpected endings.

The results of battle are too often to be found on the tails of the distribution rather at the mode. Warfare commanders should be very concerned about the 'tails' since it is there that one finds the catastrophic failures, such as the Japanese attack on Midway, and the great victories, like Napoleon's Jenna, where his supporting attack destroyed the Austrian main body. A warfare model that provides combat leaders with some understanding of the processes that drive battlefield results out to the tails would be of obvious value. Agent-based simulations offer the promise to attend to some of these deficiencies and deserve to be explored as a method for modeling warfare, of which MOOTW is a special case.

Another problem with attrition rates is that they fail to capture the mechanism by which casualties occur. The goal of any tactic in MOOTW should be, while meeting the constraint of accomplishing the mission, to minimize casualties. This should allow for casualties to be zero.

Any model that includes the use of attrition rates - Lanchester models, for example - defeat this purpose, since they assume *a priori* that casualties will be non-zero. In the "CNN era" casualties can have a profoundly negative effect on the overall success of a MOOTW campaign. Models of MOOTW combat should be able to represent the method by which casualties are generated. The understanding of this process could lead to the development of methods that interrupt or degrade it.

F.   SCOPE OF THE THESIS AND METHODOLOGY

The methodology used in this thesis represents a first attempt at combining agent-based modeling with discrete-event simulation to address a MOOTW problem. Through

11

modeling a crowd control scenario, this thesis seeks to develop a simulation methodology that will assist researchers to better understand the MOOTW combat processes.

While this thesis will ultimately simulate a specific crowd control problem and delve into the question of tactics development, there is much ground to cover on the method of modeling. In this new field the experts are few, they do not all agree, and all of the specific implementation methods are closely held intellectual property. So in a sense, this thesis is covering new ground. Thus, before addressing the specific problem to be modeled, this thesis must develop the modeling methodology. In doing so, more general application of the methodology must be considered, both from a theoretical and implementation perspective. This theoretical aspect is addressed by Holland's agent model.

Chapter II of this thesis will address the MOOTW focused implementation of Holland's model called AgentKit. Chapter III addresses the problem to be simulated, the operational questions the scenario raises, and the methods used to generate the simulation. In Chapter IV, the results of the simulation are analyzed and discussed. Chapter V contains the conclusion and recommendations.

II.   AGENTKIT

AgentKit is a software implementation based on Holland's (1995) description of a reactive agent with focus on MOOTW simulation. Before describing this implementation we need first to understand the functioning of Holland's agent model. AgentKit's structure and its most important classes are discussed. The chapter concludes with a description of the process of generating an agent-based simulation with a tool like AgentKit.

A.   HOLLAND'S AGENT MODEL

Holland (1995) provides an extensive description of his structure for a rule based adaptive agent. While the agent model employed in this thesis is founded on this theoretical model, actual implementation led to some deviation from the original design. The differences lie in the ultimate purpose of the agent models.

Holland's agent was designed to develop computer models of complex adaptive systems. Holland's goal was to allow for the agents to be able to evolve new rules via genetic algorithms. The agent we seek is a reactive agent suitable for modeling a participant in a MOOTW small-scale encounter. For a better understanding of the agent design used in this thesis, a brief description of the applicable portions of Holland's model is provided here.

Holland's agent contains four components: a set of detectors, a set of effectors, a set of stimulus-response rules, and a performance system. The agent perceives its environment through its detectors. An event in the environment causes the detectors to generate a message that are routed to all the rules. The rules in turn may or may not generate a message. The rule generated messages constitute instructions to the effectors to take a specific action in response to the event in the environment.

Holland provides the example of a frog agent. A fly approaches the frog. Its detectors (the eyes) discriminate that a small, flying thing is approaching. This message

is delivered to all of its rules for action. The only rule that provides a message is the rule that tells the tongue effector to stick out.

Holland's simple example aside, there is usually more than one rule competing for each of the effectors. The performance system is needed here to filter these messages, choosing the messages from the "fittest" rules to send to the effectors. Fitness is determined by the relative success of the rules in past decisions.

The stimulus response rules are simple IF-THEN statements. Holland stipulates a specific syntax for the rules such that the syntax is common to all agents, allows for all interactions between any of the agents, and allows for adaptive modification. In order to illustrate this syntax he proposes that the agent operate on bits throughout. The detectors discriminate only one aspect of the environment and generate a bit string of responses corresponding in length to the number of detectors. The number of unique messages possible for n detectors would then be 2^n. A rule fires its own bit string of instructions only on receiving an acceptable bit string.

To allow for non-unique bit strings he lets rules to be indifferent to the bits at certain positions in the string, illustrating this via the pound symbol (#). This allows some rules to be specific to the point of uniqueness, while others to be nearly universal. If the acceptable input of the rule is (1010101), the rule's allowable input is specific to the point of uniqueness. A rule with the acceptable input (1######) will accept any message that starts with a 1, including the previous rules message, for a total of 64 messages - half of all possible messages for this syntax. Assuming a variety of messages, this general rule will generate response quite frequently while the previous rule will rarely provide a non-null response.

Given that many rules may fire instructions to the same effector, Holland gives the Performance System the ability to discriminate between rules via two hierarchies: rule specificity and rule fitness. Assuming a varied environment, general rules will tend to fire a response more often than specific rules. General rules are valuable because of their flexibility. They can provide a response to a broader array of scenarios.

Specific rules deal with fewer scenarios but are more attuned to the few events that trigger them. The responses of specific rules are considered of greater value to the agent and are granted priority over the responses of more general rules.

The figure below illustrates the process. An event triggers the detectors. The detectors send a message that is routed to all the rules. The Performance System chooses the non-null response from the most specific rule to send to the effector. The effector accomplishes an action on the environment. The results of this action may constitute another event in the environment that starts the cycle over again. Feedback from the results leads to credit assignment. It should be noted that this agent has only one effector.

Environment

Detectors

Sends
Msg
10001

Rules

A. If 10##1
then 00100

B. If 10001
then 10110

C. If #00##
then 11110

Performance System

Effector

Msg
10110

Effector

Event

Action

Agent

Feedback leads to credit
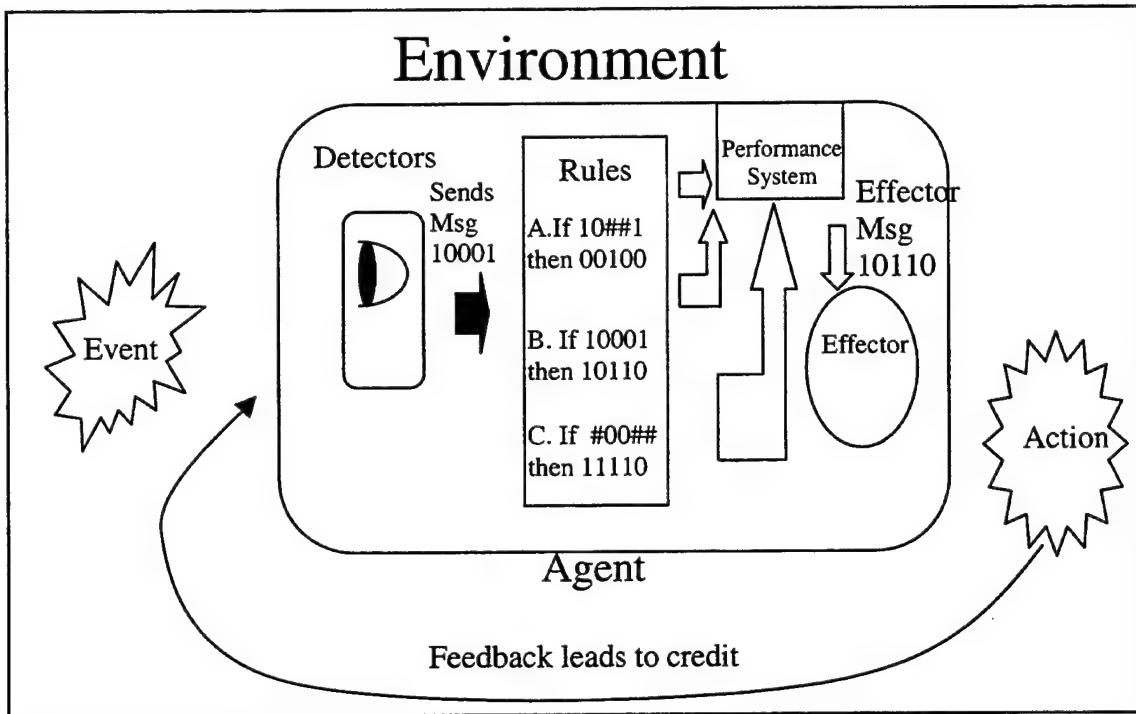
Figure 1. A graphic of Holland's agent and how it interacts with its environment. Note how all the rules respond to the incoming message and have responses. Rule B is the most specific so its response is chosen for the effector. Implied in the feedback loop is that the detectors have to detect the action in order to invoke credit assignment.

For responses from rules within the same level of specificity, the Performance System discriminates between rules by their relative fitness. The idea is to identify the rules that generate the most success for the agent. Holland ascribes to the rules a numerical weight that reflects their relative fitness. The rule whose response is chosen (highest specificity and then highest weight), is designated the active rule. The

Performance System, using a credit assignment scheme, measures the level of success of the use of this rule and increments - or decrements - the rule's weight accordingly. The more successful the rule, the more it gets used.

The total set of rules can be broken down into subsets that each map to a specific effector. Within these subsets, rules compete by specificity and fitness for the attention of their effector. This allows the processing of events from the horizon in parallel. An event triggers the detectors. The detectors' message gets to all the rules. Within each subset, the rules compete for the effector. The performance system screens these and sends the best message on to each of the effectors.

One of the strengths of this model is this inherent parallelism. A single perceived event can result in various rules being invoked that compete in parallel for the chance to deliver messages to their specific effectors. This constitutes a form of parallel decision making. Thus, an agent model of a soldier that started receiving enemy fire could make the decision to return fire, call out a warning, and run for cover, all without having a single rule that specified this response. In contrast, a traditional Artificial Intelligence (AI) method would search the available database given the current states and select the matching rule.

Parallelism allows a building block approach to rule building (Holland, 1995). One way to look at this approach would be to envision the rules whose responses arrive at the effectors as being a one meta-rule output, composed of the parallel rules' outputs strung together. Then, a rule set of m rules per each of n effectors (a total of m*n rules) could minimally describe an m^n meta-rule set. In actuality, the number in this set is probably higher when one considers that some rules are general and handle more than one situation.

Figure 2 provides an illustration of this process. The picture shows three effectors and their respective rule sets. Some event in the agent's environment has led the performance system to deliver a rule response to each effector. If we were to consider a sequential treatment of this process, we would need one rule to represent each combination of three effector rules. Thus, the 12 rules pictured deliver as many possible responses as a sequential system of 81 rules.
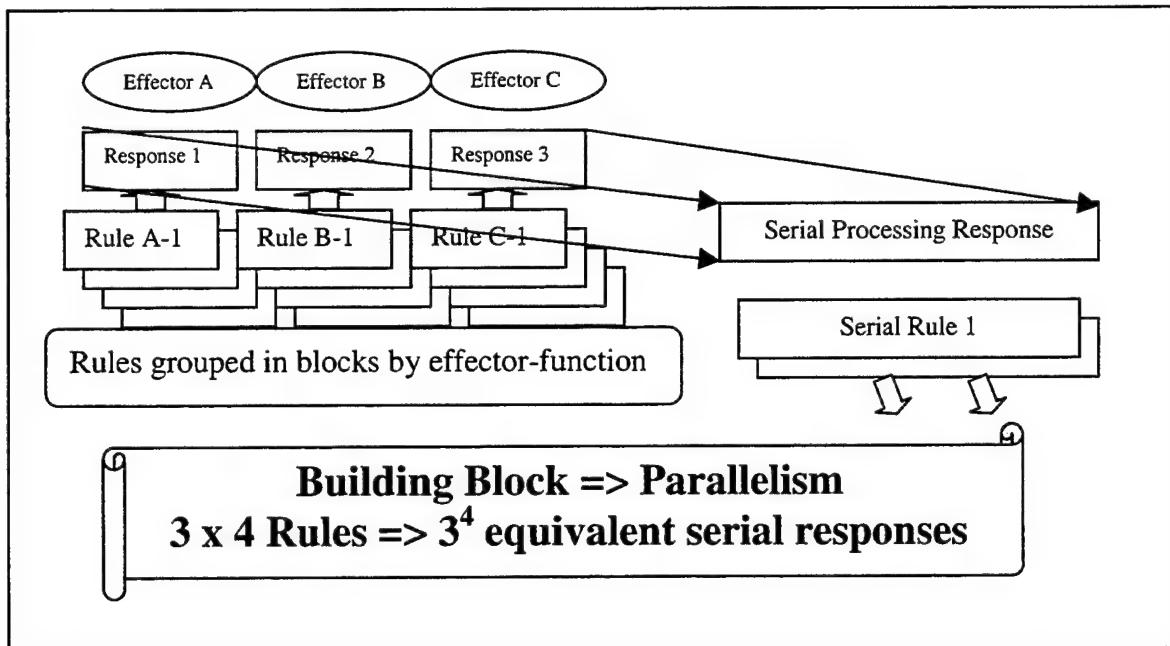
Figure 2. Comparing the Parallel Process effect of using rules in building block fashion versus using serial processing rules.

Parallelism, rule competition, and the building block approach allow Holland's agent to model agent adaptivity. Credit assignment creates a competitive landscape for rules within the agent. It is a first order analogy of individual learning: creating models, testing them, discarding those that work, and retaining the successful. The challenge lies in deciding the details of what events equate to success or failure and the weights assigned for these event - what Holland calls the credit assignment scheme.

The final step in Holland's scheme for agent adaptation is Rule Discovery. This involves evolving new rules from experiences gained by the agent. Here Holland applies genetic algorithms. The rules can be thought of individual functions whose mapped value on the fitness landscape is their weight. The rules are defined by the bit-streams that form the input/output. Holland uses these bit streams as the genetic code on which he applies crossover and mutation to generate new rules.

The reader may have noticed by now that nowhere in the description of Holland's agent does the term "goal" arise. This seems in direct contradiction to Ferber's

description of an agent, though in all other aspects, Holland's agent fits the bill of what Ferber calls a reactive agent. Looking deeper, one can note that Holland's agent is driven by implicit, internal goals that lie within its credit assignment scheme - the agent's method for ranking its internal models. For instance, an agent model of an animal has as its goals to survive and reproduce. In Holland's agent there is no explicit rule or method that says "I must survive" or "I must reproduce". It is implied, since the credit assignment scheme rewards the rules that allow it to eat, survive threats, compete, and reproduce.

B.   IMPLEMENTATION CHOICES

The first step in implementation was choosing a programming language. Given that Holland's agents were essentially constructs of component objects, an object-oriented language was preferred. It was desired that the implementation be platform-independent so simulation models using this implementation could be run on nearly every machine and operating system. Java is the language that meets all these criteria. Additionally, Java is the language used to write Simkit (Stork, 1996), the discrete event step (DES) simulation library employed as the foundation for AgentKit.

The DES part of simulation was provided by Simkit: event list, simulation clock, random variable generation, data collection, and generic simulation entities. Simkit also contains some more specific classes of objects such as Mover, Sensor, Mediator, and Referee, that allow for the simple modeling of physical scenarios. With little work, the user can create new objects with the desired functionality by extending the abstract SimEntityBase class in the library, or by extending the more specific objects mentioned above.

Thus, by using Simkit as the simulation environment for Agentkit, the task of implementation of Holland's agent was one of extending Simkit objects and granting them the necessary attributes. The majority of the objects used to form AgentKit agents are extensions of Simkit objects.

C.   AGENTKIT STRUCTURE

A useful and effective implementation for agent-based simulation of MOOTW demanded a general purpose, component approach. This served two purposes. First, following this approach allowed for the design implementation to progress in phases. Trying to implement Holland's concepts required breaking the concepts into layers of complexity. Trying to "eat the elephant in one bite" would result in a hard-coded, inflexible design that would be fragile and difficult to modify as well as opaque to follow-on researchers. Thus, the design progressed from the very general to the very specific. Second, no design is ever perfect and a robust component framework allows for easy modification and improvement, both during the progress of this thesis, and in the event of follow-on work.

AgentKit consists of two parts: the Foundation library and the MOOTW library. The Foundation library consists of mostly interfaces and classes generic to AgentKit. The MOOTW library contains classes designed for MOOTW simulation.

D.   FOUNDATION LIBRARY

An agent in AgentKit is assembled from various objects. The kernel of the agent is composed of objects derived straight from Holland's agent: a set of effectors, a set of detectors, a set of rules, and a performance system that manages the interactions.

The detectors perceive the environment and generate messages about their perceptions. These messages are delivered to the rule set. The messages may or may not trigger individual rules. Rules that govern specific effectors compete in some sense to deliver their messages to the effector. The effector receives a message and executes an action. The performance system oversees the messaging and adjudicates which rules get to send messages to the effectors by choosing the non-null response from the rule of highest specificity.

In AgentKit, the roles described above are filled by software objects from the following interfaces and classes.

INTERFACES	DESCRIPTION
Effector	Object that acts on agent's environment
Detector	Object that senses in agent's environment
Rule	Provides an output message to appropriate input
MessageGenerator	Used by Detector for sending messages
CreditHandler	Assigns credit to Chromosomes

CLASSES	
PerformanceSystem	Object that fulfills role of a performance system
Gene	Bundle of rules governing a specific Effector
Chromosome	Bundle of genes that map to the Effector set
MessageEvent	Message and Originator data bundle
Message	Wrapper for an Object array

The PerformanceSystem can be viewed as the neurological system of the agent. It routes messages from the detectors to the rule set. It polls the rules for their responses and then selects responses to forward to the effectors. An instance of this class maintains references to three variable sets: a hashtable of effector objects, another of Detectors and an array of Chromosomes. The hashtables are keyed by the name of the function the effector or detector fulfills. For example, to get the reference to the agent's shooter, query the effector hashtable with the key "shooter". This allows the Performance System to remain independent of the specific class used for a specific function.

A significant difference between this agent and Holland's is that the rules have no individual weight. Instead, the agents are allowed to have separate states. Each state will have a different rule-set, the chromosome. It is the chromosomes that are weighted. The state of the agent is a function of the active chromosome - the chromosome with the highest weight. Credit assignment, instead of giving rewards to rules, is dealt to the chromosome via a scheme of the user's choice.

Assigning credit to the active chromosome vice individual rules (representing gene alleles in the genetic analog) is more in keeping with Fogel's arguments (2000). Research in genetics indicates that the individual effect of a single gene is very difficult

to isolate (Fogel, 2000). A specific gene on one chromosome may produce entirely different traits in the phenotype than the same gene on another chromosome. Thus, the active chromosome is rewarded or punished based on the user's credit assignment scheme.

In terms of structure, a chromosome is an array of Genes. A chromosome maps each gene to an effector in the effector hashtable. When the Performance System queries the Gene with a message, it is understood that the responses it is gathering are competing for the attention of a single effector. A Gene object is an array of Rule objects sorted in descending order by the rule specificity. If a rule in a gene has no response to a query it returns null and is ignored by the performance system. The performance system accepts the first non-null response (since the more specific rule has priority) and feeds this to the appropriate Effector.

In AgentKit, the agent can be modeled as simply reactive by giving it only one chromosome in its chromosome array. Alternatively, adaptivity is modeled simply by providing the agent with many chromosomes. The active chromosome is determined via a credit assignment scheme of the user's design.

The performance system will always provide every chromosome with incoming messages. However, it will only forward responses from the active chromosome to the effectors. This allows for the use of rules that maintain state, should the user desire.

Take the case of an agent modeling a combatant with two states: hostile and passive. When the agent is in its passive mode, it has a rule for fleeing that is invoked by taking enemy fire. Suppose it is in the aggressive mode taking fire when something triggers it to shift to the passive mode. It makes no sense for the agent's flee rule to be unaware of the fact that the agent has recently taken fire.

A simple way to address this is to allow the flee-rule to be aware of the agent's situation by tracking all incoming messages. Thus, while the agent is in the aggressive mode taking fire, the flee rule will be sending flee messages the entire time, but these messages will be ignored until such a time as the agent enters its passive state.

The effector and detector interfaces allow the user to use previously designed objects and extend them, while implementing the necessary interface for use in AgentKit. For instance, the Simkit library class BasicMover was extended as

BasicCombatAgentMover implementing the interface Effector. Thus, a Mover type object could also be treated as an Effector.

The Detector sends messages via the MessageGenerator. Having a separate object for message generation keeps the method of detection orthogonal to the messaging method.

Rule objects also handle messages but rather than generating messages that are sent to the performance system, their messages are handed directly to the performance system in response to the messages provided.

E.   MOOTW LIBRARY

The MOOTW library contains over 40 classes, so the discussion here will focus on the eleven most important ones. All of AgentKit's classes are listed in the Appendix.

CLASSES	DESCRIPTION
BasicCombatAgent	Wrapper for all the objects that make an agent
BasicCombatAgentMover	Movement component for a CombatAgent
BasicShooter	Object that has weapons and fires them
BasicVisualSensor	Cookie cutter sensor
AgentReferee	Tracks all movers and oversees interactions
CombatAgentMediator	Mediates interactions between two movers
FireArm	A simple firearm - needs its parameters set by user
MoverRule	Abstract rule for movers
ShooterRule	Abstract rule for shooters
ModularMoverManager	Manages the moving process as per the active rule
ModularWeaponManager	Manages the shooter process as per the active rule

The BasicCombatAgent serves as repository for all the components of the agent: chromosomes, effectors, detectors, and the performance system. The BasicCombatAgentMover allows the agent to move in its environment. The

22

BasicVisualSensor are its eyes. BasicShooter is an effector that carries out the agent's shooting functions. FireArm requires the user to input its parameters: ammunition capacity, maximum range, rate of fire, and standard deviation in meters per meter range.

Interactions between separate entities in AgentKit, as in Simkit, are handled by a third party. First, the AgentReferee tracks all the entities. When it determines by their trajectories that there will be an interaction between two entities, it assigns a CombatAgentMediator to handle the specific interaction. The CombatAgentMediator, an extension of Simkit's Mediator class, handles detection by informing the sensor when it has detected the mover, contingent upon the sensors detection algorithm.

The CombatAgentMediator also handles other interactions: enemy and friendly weapon firing, near misses, target kills, etc. For example, when the BasicShooter discharges his weapon, the CombatAgentMediator listens to this event and uses the Rayleigh distribution (with the standard deviation from the weapon scaled by the range to the target) to calculate probability of hit. This mediator then informs the target that he was either hit or had a near miss. Figure 3 provides a caricature of the process. The CombatAgentMediator detects Tank A firing, calculates the probability of hit, generates a uniform random sample, and determines that Tank B was hit.
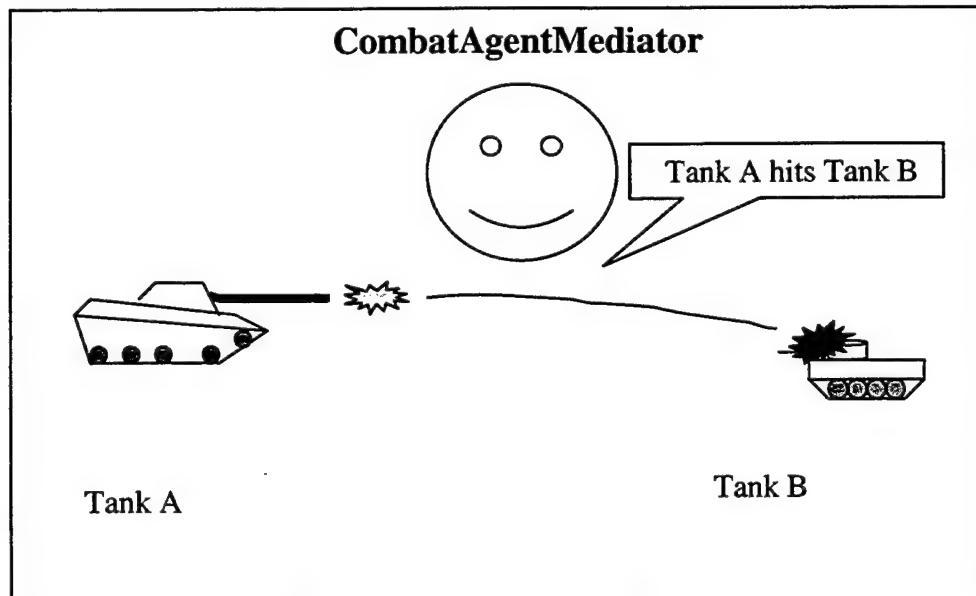

Figure 3. The CombatAgentMediator's role.

The default damage function is simply the number of hits. The BasicCombatAgentMover can a take an arbitrary number of hits (set by the user) until dead. He suffers no degradation in capability until dead. The user, however, may specify a more elaborate scheme for a damage function.

MoverRule and ShooterRule are abstract classes. They do not provide any kind of movement algorithm but do contain all the variables and methods one would need for rule "housekeeping". To implement a desired movement algorithm, the user extends the applicable abstract class and encodes the desired movement or firing algorithm.

In Holland's model the rules are one-dimensional. They make a decision, firing a response to an input implemented by the effector. However, in discrete event scenarios things get more complicated. Events can arrive very quickly and if every event triggers a decision, the agent may be stuck in a repeated cycle of event-message-decision, ultimately do nothing. Conversely, some rules may require multi-step tasks; they may be algorithmic rather than simple IF-THEN statements. A rule to pursue a fleeing and evading foe is arguably a non-linear search problem. In AgentKit, a MoverRulePursue class handles this by linear steps each a fixed proportion of the distance to the target. By allowing the rule to terminate within a fixed distance of the target, we are assured of finite convergence of the algorithm. As the reader may note, this requires some method of communication between the rule and effector, something Holland's model does not allow.

AgentKit designates a third party called a modular manager to go between the effector and the rule. The modular manager keeps track of the active rule and informs it when the effector has completed a process step and is awaiting instructions. The rule returns a new instruction and awaits the next request. This cycle continues until supplanted by a new active rule. This interaction between the modular manager and the rules permits the use of algorithmic rules. This idea of allowing algorithmic rules is one of the more significant differences between AgentKit and Holland's model.

So far, we have discussed the components of agents in AgentKit. At this point it is important to discuss the process of creating agent models with AgentKit.

F.   THE AGENT MODELING PROCESS IN AGENTKIT

As described previously, Axelrod (1998) describes the process of research with agent-based simulation as being a combination of inductive and deductive learning. The agent-based modeler uses deductive logic to create the agent model and the rules that govern them and from his simulation he or she gains inductive insight.

AgentKit's function is to facilitate the deductive portion of modeling the agent. AgentKit provides the components while the modeler fits them together to make the individual entity that suits his or her modeling needs. The agent modeling process used in this thesis consisted of the following:

- Choose detectors that describe how the agent gathers information from the environment.
- Choose effectors that model how the agent affects its environment.
- Devise a simplifying rule set that adequately describes the agent's real world behavior. Start with simple general rules and incrementally introduce specific rules.
- Devise a message format so that all messaging can be understood. The sender does not always know nor care who the receiver is. For example, detectors send detection messages for all the rules to read.
- Describe the set of events that interact with these rules. This plays a key role in what the detectors should be able to "see" and what the Mediator mediates.
- Consider the need for individual adaptation (evolution), because it may not be vital to the model.
- Add rule and/or functionality incrementally, interspersed with extensive testing to determine if the individual behavior of the agent is satisfactory.
- Test/adjust the agent to see if its individual behavior adequately captures the desired aspect of the entity being modeled.

For MOOTW, although the entities being modeled by agents are people, the roles they play help narrow the number and types of behaviors that will have to be captured.

Detectors can be simplified to just the eyes. Effectors may be only a movement component and a firing component.

A richer model could include a communication component, an ability to take possession of and handoff objects, or a role-playing component, such as leadership. The user needs to exercise care in choosing the number of features to add to the agent. The model should be built up incrementally in order to minimize the features of the model to avoid unnecessary complexity.

Rules are needed to govern these effectors. For example, rules for movement need to reflect different situations: following a leader, moving to a desired position, fleeing a foe. Again, while it may be attractive to build a rich and diverse rule hierarchy to capture every possible behavior, the modeler must employ caution in choosing these rules. The rules are driven by the events that the modeler wishes to incorporate in the model. By deciding on which events are important to the MOOTW scenario and discarding the rest, the modeler will also constrain the types of rules he or she may employ. For example, initial detection of other entities is a mandatory event but will the model include detection of rioters throwing rocks, or a fellow peacekeeper being hit? If the model includes detection of rioters throwing rocks, then rules dictating reactions to this event can be used.

With this understanding of the process that generates an agent-based simulation, we may proceed to the following chapter where we discuss the riot control scenario addressed in this thesis and the simulation model.

THIS PAGE INTENTIONALLY LEFT BLANK

III. THE RIOT SIMULATION MODEL

This chapter presents a riot control problem faced by U.S. Soldiers and Marines in the Cuban migrant camps at Guantanamo Bay, Cuba, in 1994. A simplified scenario is constructed that generalizes the riot control problem. The scenario is then simulated using AgentKit.

A. THE RIOT SCENARIO

During 1994, a large number of Cubans attempted to immigrate to the U.S. illegally. Many were undesirable members of Cuban society. Several thousand were detained and kept in large immigrant camps at the U.S. Naval Base, Guantanamo Bay, Cuba. There, in August and September 1994, the Cuban migrants rioted violently. The security forces at the base consisted of a Marine infantry battalion and a U.S. Army Military Police force. It took the security forces five days of sustained civil disorder operations to stabilize and quiet the migrant camps. LtCol. John R. Allen wrote a Marine Corps Gazette article about the part played by 2nd Battalion, 6th Marines, which he commanded, during this operation and the lessons learned. This article provided the inspiration for this thesis. (Allen, 1995)

LtCol. Allen describes the initial encounter with the raw violence of a rioting crowd as being very disconcerting to the Marines and soldiers involved. Leaders were taken aback at the newness and unexpectedness of the behavior presented. Neither their training nor their doctrine had prepared them for the initial encounter. They had to adapt their skills to the situation at hand and develop new tactics to counter the advances made by the crowd, in effect inventing new tactics on the spot. The collective body of knowledge regarding MOOTW combat has advanced because of their experiences and those of other units in similar situations. However, exploration of future MOOTW encounters is necessary. Field tests, like those conducted by the Marine Corps Warfighting Laboratory are useful.

This scenario could be viewed as a template for MOOTW violent encounters. Developing an agent-based simulation of this scenario could provide valuable insights

28

into the challenges of MOOTW small-unit combat and the process of casualty generation. It could provide us a valuable model that could enable the process of tactical development. It is the author's opinion that the true potential of agent-based simulation of MOOTW combat is not in terms of generating predictions for the success of Tactic X versus Tactic Y, but in generating insight and understanding as to why Tactic X may be better or worse than Tactic Y.

B.   A GENERALIZED SCENARIO

The idea of tactics development via agent-based simulation will be tested on an abstraction of the scenario described above by LtCol. Allen. A small force armed with imperfect non-lethal weapons in a congested urban area guards a high value site. Their implied mission is to minimize violence. Facing them is a large, lightly armed force seeking to exert violence and seize the valuable site. Assume that several tactics have been developed for the force in this situation and require testing. The tactics will be evaluated using an AgentKit model. To assure completeness, the tactics should be tested against crowds of varying composition. Juxtaposition of tactics against crowd composition can be studied and evaluated using experimental design with the AgentKit model.

The mission of the peacekeeping force is to bring order to an urban zone torn by civil disorder. The local population is driven by profound discontent with the current situation. Throughout the urban zone there are many likely targets upon which the local population is likely to focus its destructive potential. Examples include a bridge crossing into the opposing ethnicity's district, a supermarket to loot, and an exit gate from a large refugee camp.

The peacekeepers, having limited assets, deploy small detachments armed with imperfect non-lethal weapons to guard each of these sites. These weapons are "imperfect" in the sense that while they weapons were designed to be non-lethal, to cause an individual to lose the will the fight, they have nevertheless been known to produce lethal effects (Kenny, 2000). The detachments are supported by reinforcements and support elements that can arrive in a fixed amount of time. The policy for the

peacekeepers is not to wait for violence to begin but to call for support as soon as they notice a crowd forming.

The crowd may present itself in two forms. The first, the Homogenous Crowd, forms spontaneously without discernable leadership. That is not to say they are not led, since they may have local leaders that arise from the group. However, it does imply that there is no organized leadership.

The second form, the Heterogeneous Crowd, does have a leader. The leader is a motivator who arouses the crowd to violence. He does not himself actively participate in the violence but continues to spur the crowd onward.

The crowd is armed uniformly with manually thrown projectiles, such as stones, sticks, or bottles. Individuals by themselves are not prone to commit violent acts but when they gather into a crowd, they reinforce each other's anger over time, leading the aggregation to transition from a protesting crowd into a violent riot. The crowd's goal is to get to the valuable site.

The peacekeepers have a choice of two tactics to employ should they encounter a violent crowd. The first tactic is called the Reactive Tactic. This tactic allows the crowd to approach but, if it turns violent, returns violent acts tit-for-tat. If a rioter throws a stone, and is seen, the peacekeeper fires one non-lethal round in return. It is hoped that even should the peacekeeper's shot miss, the rioter would be dissuaded for throwing another rock.

The second tactic is the Proactive Tactic. Here the policy empowers the peacekeepers to make assessments of the crowd and their potential for violence. Should a crowd begin to demonstrate violent potential, i.e. picking up projectile weapons and closing range on the guarded site, the peacekeepers may shoot at the member of the crowd with the greatest potential for violence once the crowd enters stone throwing range. Implied in this rule is the idea that crowd agitators, even if not themselves armed, may contain the greatest potential for generating crowd violence (Allen, 1995).

This tactic actively targets leaders. Leaders, as mentioned earlier, can be the agitators or the first-to-act crowd member whose actions can serve to trigger others to action. It is hoped that this tactic will prove capable of dissolving the crowd before it becomes excessively violent.

30

The peacekeeper detachment's mission is to guard the valuable site, but in this, they have two other large concerns: to ensure their own survival and, to use the minimal force necessary to control the crowd. There are two measures of success, one primary, and the other secondary. The primary measure is their survival until the arrival of the support and reinforcements. Thus, if they survived the confrontation, it meant the site was successfully defended. This implies that they will defend the site with their lives (possibly an unrealistic assumption). The second measure is the amount of damage inflicted on the crowd. Obviously, if there is low damage to the crowd but the peacekeepers are dead, this measure is meaningless. On the other hand, to completely minimize damage to the peacekeepers, the solution that maximizes damage to the crowd at first sight is just as poor. A balance must be struck between the two extremes.

This scenario will be modeled as a simulation using AgentKit. The simulation will be used to conduct a two factor, two level experiment. The two factors are leadership and tactics, and each has two levels.

C.   GENERATING THE MODEL

Having established the scenario, the next step is to model it. We will apply the techniques of modeling CAS with agents to formulate the above scenario in AgentKit. The individual participants are modeled as agents with only the necessary characteristics that capture applicable individual behavior. A set of rules governing the "physics" of the simulated reality are generated. The agents are then placed into this simulated reality and allowed to interact.

The simulation places a detachment of peacekeepers arriving at the guarded site as a crowd is beginning to form. Both groups are within visual range of each other but beyond range of the rioters' weapons. The important parameters and interactions of the simulation are contained in Table 1.

The entities within the simulation are all as instances of the class BasicCombatAgent. It behooves us at this point to gather a better understanding of the specifics of this agent model.

31

Parameter	Method	Mean/Fixed Value	Distribution	Other
Run duration	Deterministic	7 minutes	Na	At 7 minutes support shows up.
Movement	Constant speed	Set by user	Na	Movers moved at max speed or 0; i.e., instantaneous acceleration
Weapon inter-firing times	Deterministic	Firing rate input by user	Na	Example: Non-lethal weapon fires 8 rounds per minute.
Hits	Stochastic	Dependent on range	Rayleigh	Each weapon has a standard error per meter range.
Lethality	Deterministic	Hits until dead input by user	na	Rioters took 2 non-lethal hits to die; peacekeepers took 5 stones to die.
Communication	No overt	Na	na	Rules regarding firing and movement included behaviors based on implied communication
Sensing	Cookie-cutter	Input by user	na	Assumed congested environment: 100 meters.

Table 1. Riot Simulation features and interactions

D.   BASIC COMBAT AGENT

BasicCombatAgent is the AgentKit entity that serves as the foundation for all agents used in the simulation. It serves as a repository of all the components that make the agent: performance system, the effectors, the detectors and the chromosomes. The detectors set consists of only a cookie cutter sensor. The effector sensor consists of a mover, a shooter, a credit handler, and heat radiator.

The mover has a color and is shaped like a sphere; i.e., it presents a circle of specified radius to all directions. The shooter possesses a weapon and fires it on command from its rule set provided the weapon has remaining rounds. Both the mover and shooter possess an object called a modular manager. This allows the active rule of the effector to interact with the object whose actions it dictates.

The credit handler allows the agent to effectively possess more than one chromosome set. For this simulation a simple credit handler was devised that set the credited chromosome to weight 1 and all others to 0. The method of awarding credit is independent of the method of determining what are credit events. Thus, if a user wishes to retain a credit scheme but to award credit differently, one has simply to chance the credit handler. The heat radiator will be discussed as a feature of the Rioter.

32

Damage to the agents is handled deterministically for simplicity's sake. The user may set the number of hits to the agent can take before dying, or this parameter at the default value of one. Alternatively, the user can devise his or her own damage function. For this simulation, damage was calculated by DamageAssessor, a static class with references to all the weapon classes. When hit, the agent determined what weapon had struck it and assigned the number of damage points given for that class weapon by Damage Assessor.

As mentioned in the previous chapter, the BasicCombatAgent's rules are found in the chromosomes. Each chromosome contains a set of genes with each gene mapping one-to-one on the agent's effectors. The genes in turn hold one or more rules governing their effector. In AgentKit, an individual gene will hold only rules of differing levels of specificity. The user provides the combat agent with its rules after instantiation.

While all agents are BasicCombatAgent instances, they are differentiated by the roles that they play. An agent's role is a function of his rules and resources. The roles are Rioter, Riot Leader, and Peacekeeper. Their relationship to BasicCombatAgent is pictured in Figure 3.
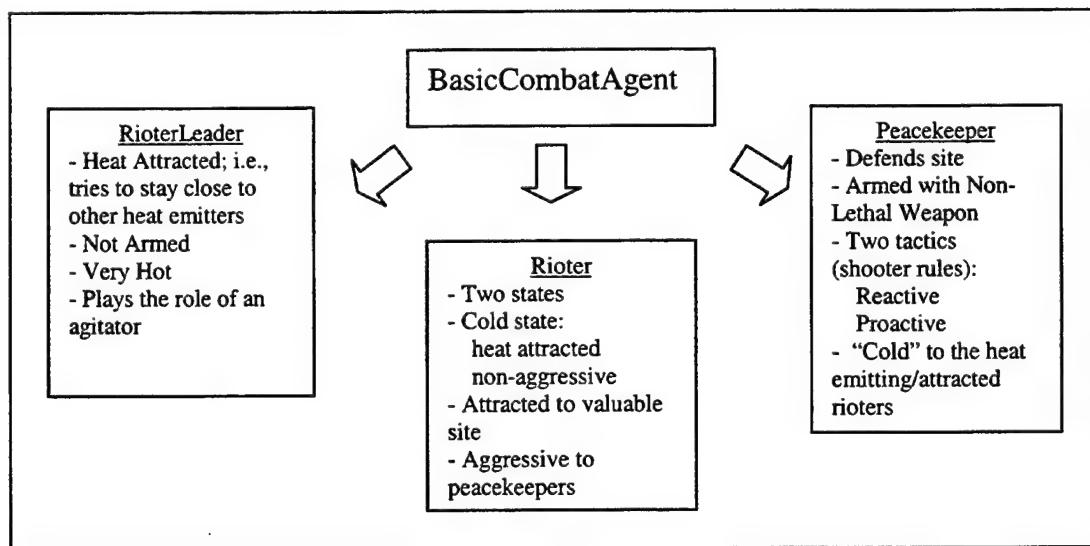

Figure 3. BasicCombatAgent and the roles it fulfills in the simulation.

E.   THE RIOTER

The rioter is a BasicCombatAgent but provided with rules and attributes that make it behave like a rioter. The rioter is supposed to behave like a crowd member in that it acts scared when alone, but finds strength in numbers. When enraged, her/she throws rocks at the peacekeepers. Modeling the rioter required finding a way to have groups of them mimic the behavior of a rioting crowd. The first step was to look at the individuals in the crowd as being thermodynamic particles possessing different states as a function of their temperature. The rioters generate, emit, and absorb heat via their heat radiator effectors. Groups of rioters start in a cool state but become "hotter" the longer they congregate until individuals "boil over" into a hot, violent state.

What then are the "thermodynamics laws" of this simulation? Heat absorption, loss, and emission were adapted from physics. The temperature scale is continuous with range $[0, \infty)$. The agents, in the absence of another adjacent heat source, generate just enough heat to maintain a temperature of 10 degrees. They lose heat via emission in accordance with Stefan's Law (proportional to quartic of the temperature). The Stefan-Boltzmann's proportionality constant was replaced with a tuning factor to reduce the amount of heat lost at each time step. The heat the agents receive from other agents is in inverse proportion to the square of the range. Thus, the rioters only got hot when several got in close proximity of each other. (Sear, 1976)

Further, a tuning factor was used to control the proportion of absorbed heat the agent was allowed to use. In essence this tuning factor served as a measure of the crowd's acceleration towards a boiling point. This meant that the small "crowds" actually used in the simulation could be made to riot. Another use for this tuning factor was to model the desire to avoid the peacekeepers. By making this factor negative, the peacekeepers became heat sinks for any rioters close to them.

As mentioned earlier, a rioter has two states: passive and aggressive. The rioter's current state is a function of its temperature. A rioter's implied goal is to maximize its temperature when in the passive state and when in the aggressive state to keep it above the "boiling point". An arbitrary temperature of 100 was set as the state transition point. When at a temperature in the range [10, 100], the rioter was governed by a drive to move

towards heat. In this state they were non-aggressive to the peacekeepers. Once above a heat level of 100, their movement rule was to move towards the high value site. The firing rule was to throw rocks at the nearest peacekeepers.

Rioters engage peacekeepers with rocks. RockThrower, the object that represents this function, has a firing rate of 12 rocks per minute with a maximum range of 50 meters. The rock impacts are modeled with a Rayleigh distribution with a rate of 0.033 meters per meter range. This puts the p-hit against a circle of 0.5 radius at 50 meters at 0.045, or roughly a one in 22 chance of hitting a man's torso at 50 meters. (Wagner, 1999)

The rioters can take one hit from the peacekeepers non-lethal weapons with no degradation in function. However, since the heat radiator also plays the role of a credit scheme, where the measure of credit is the current temperature, then being struck is a negative credit event. Thus, one hit brings the rioter's temperature down to 10 degrees, causing it to become passive again and invoking its flight rule.

Being hit also degrades the rioter's ability to absorb heat by reducing the heat-absorption tuning factor so that it will take them much longer to warm up. A near miss causes a heat drop of a tuning factor multiplied times the damage points of the weapon class. If an agent is barely above the boiling point then a near miss will cause it to shift to passivity again. However, if the rioter is very hot, he will likely remain above the transition point but be vulnerable to any additional near misses.

F. THE RIOT LEADER

The riot leader is essentially a rioter stuck in the passive state. It is never explicitly aggressive towards the peacekeepers. It is always heat attracted. However, its heat radiator object is different than the rioters in that it radiates intense heat continuously. It tries to maintain its temperature at 500 degrees. Thus, it mimics the agitators that LtCol. Allen discussed. It acts as a catalyst in precipitating the "boiling over" of the crowd.

The riot leader "dies" at one hit. This was a crude way to model LtCol. Allen's tactic of using a snatch team to remove crowd agitators. Near misses reduce his

35

temperature in identical fashion to the rioters but in the next heat calculation iteration its temperature restarts its exponential climb to 500 degrees.

G.   THE PEACEKEEPER

The peacekeeper has a single state. It uses a defensive movement algorithm that would model the behavior of troops around a defended site. This rule requires its peacekeeper to stay between three and six meters from the defended site. The peacekeeper is attracted towards the enemy and repelled from other peacekeepers. The expected behavior is that if no enemy were present then two peacekeepers would end up on opposite sides of the guarded site. If one peacekeeper was guarding the site and there were enemy present, it would place itself between the guarded site and the enemy. With more than one peacekeeper and an enemy presence, the behavior results in a combination of these two scenarios.

The peacekeeper is modeled as having some body armor protection: a helmet, face shield, and flak jacket. This allows it to die only after the fifth stone hit. Heat is not relevant to this agent so near misses have no significance.

The peacekeeper is armed with an imperfect non-lethal weapon. This simulates a single shot rubber bullet round as fired from the M203 grenade launcher. Since there is no specific rate of fire for this round in the literature, a rough estimate of 8 rounds per minute was made based upon the author's infantry experience. The weapon was arbitrarily assigned an error of 0.011 meters per meter of range. This resulted in a probability of hit against a rioter at 50 meters, based on the Rayleigh distribution, of 0.323. The weapon's imperfect lethality is modeled by making the second impact on the same target deterministically lethal.

As mentioned earlier the peacekeepers have a choice of tactics. These represent the Rules Of Engagement (ROE) that would be assigned by higher command authority. ROE can be permissive or restrictive. The Reactive Tactic is representative of Restrictive ROE. It requires the peacekeeper to fire one non-lethal at the most recent rock thrower detected. For the purposes of the simulation a 50% error was applied to this process to simulate the "fog of war", that is, confusion, mistakes, distractions, and interrupted line of sight.

The Proactive Tactic represents permissive ROE. It allows the peacekeepers to engage the greatest threat that demonstrates immediate intent to do harm. In this simulation this meant identifying the rioters who were already above 100 degrees and within rock throwing range. The algorithm then picked the hottest rioter to engage subject to a 50% error. The error made it feasible occasionally not engaging any targets at all. This is seems reasonable, since it can be interpreted to simulate how some troops may freeze in indecision. It is, however, relatively rare. A single peacekeeper facing five violent rioters has a one in 125 chance of doing nothing. In initial test runs of this rule, one behavior noted was that the rule allowed crowds to occasionally get close and overwhelm the peacekeepers.

Looking deeper into the results and the simulation's animation revealed that even at close range (as expected) the peacekeepers would attempt to engage the hottest rioter, even if it was not the one firing at them. The rule failed to capture the nature of self-preservation actions of troops under fire. Thus, a caveat for self-protection was added to the rule, which allowed the rule to engage the hottest until a rioter came within the self-protection range of 10 meters. At this point the peacekeeper engages the closest rioter.

H.   A TYPICAL SIMULATION RUN

After extensive testing of the rules and the software, a setting was established for the simulation runs. The crowd consists of seven individuals: either six rioters and a leader, or seven rioters. The peacekeeper detachment consisted of two peacekeepers. The fixed run time was set at seven simulation minutes. The crowd begins at scattered in uniform random fashion in a 20 by 20 square around a fixed point.

This fixed point is set at a distance of 70 meters from the valuable site. Given that the rioters begin in a passive, heat attracted state, they spend the first few minutes moving closer to each other and getting hotter. As the rioters begin to transition to the aggressive state, they begin to move towards the valuable site. When within stone throwing range of the peacekeepers, they begin to throw stones. There are two kinds of results from the ensuing combat.

The most common result is a victory for the peacekeepers. These encounters unfold in a typical fashion. As rioters in the aggressive mode take direct hits from a

37

peacekeeper, their temperature precipitates to 10 degrees and they begin to flee. The same effect can occur to a rioter from a near miss depending upon his current temperature. As rioters begin to attrite, the remaining rioters begin to lose heat since they have fewer of their fellows around them. Additionally, they come under the heat sink effect of the peacekeepers. It was not unusual to observe in some of the runs, that rioters would transition to passive mode and flee simply due to heat loss.

These conditions stack the odds against the rioters. However, circumstances did arise when, through many misses by the peacekeepers, the rioters were able to close with the peacekeepers in mass. This was the problem with the original reactive rule. When the rioters achieved this condition, the higher hit probabilities due to the close range, their greater numbers, and higher rates of fire allowed them to overwhelm the peacekeepers.

For every run, the results of the combat were printed to a text file. Additionally, the state of the pseudo-random number generator seed was dumped to a file. Thus, each run was essentially independent. Each variable setting was allotted 40 runs for a total of 160 runs. Upon completion of all the runs, the text files were examined and the data manually entered into Excel files.

An animation package was used for debugging and to allow qualitative interpretation of the results. The animation package was very simplistic. A light bulb represented the valuable site and the rioters and peacekeepers were red and blue dots respectively. The riot leader was animated as a tiger head. A sample of the animation is in Figure 4.

At this point we have gained an understanding of the simulation model and the data it generates. The next chapter will provide the experiment's design, the analysis of the experiment's results, and a discussion of both these results and the methodology used to generate them.
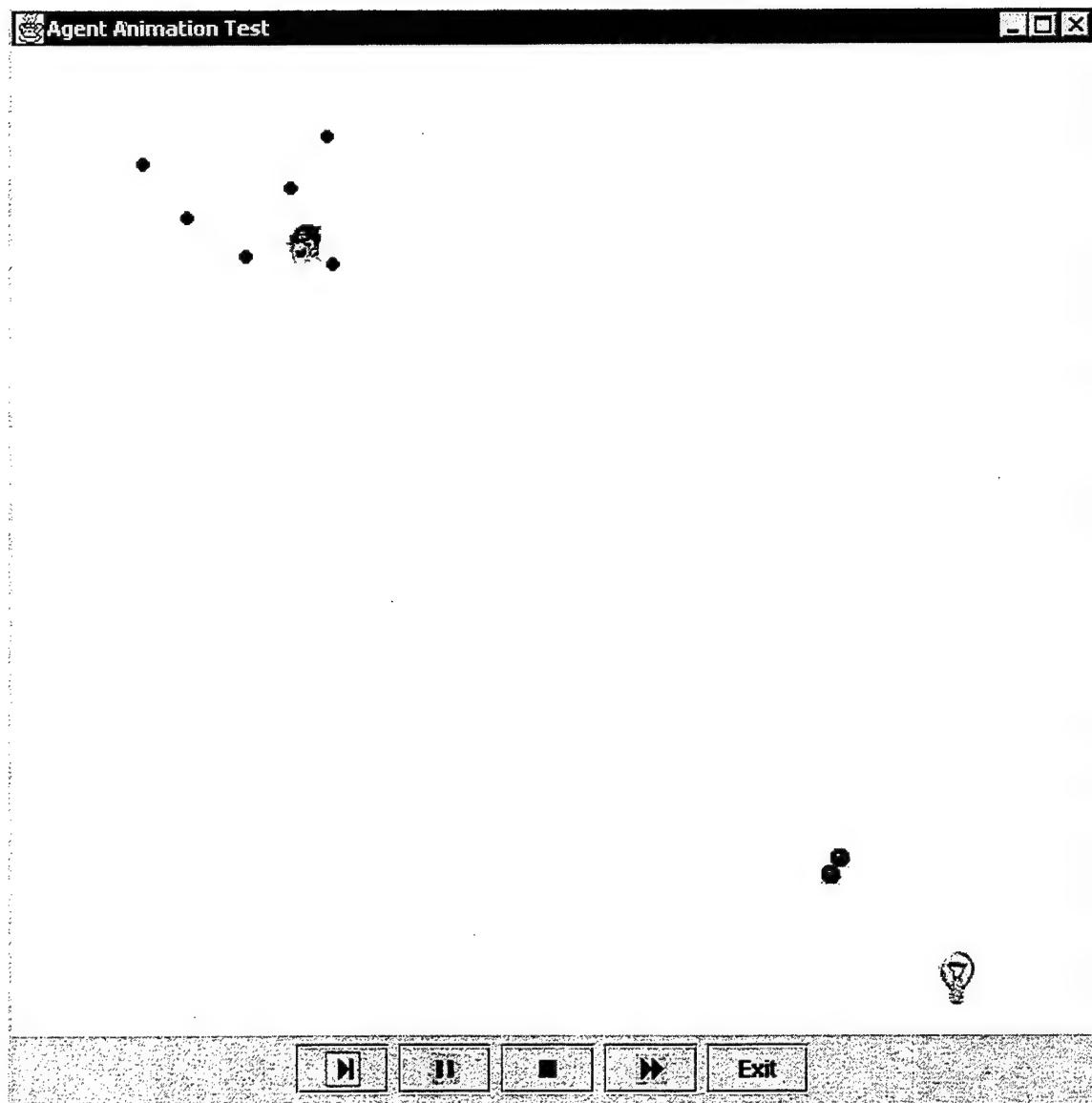
Figure 4. A snapshot of the animation of the Riot Simulation in mid-run. The two large blue dots are the peacekeepers guarding the valuable site as represented by the lightbulb. The small red dots are rioters and the tiger head is the riot leader. Not shown by the animation is the firing process.

THIS PAGE INTENTIONALLY LEFT BLANK

IV.   SIMULATION RESULTS

This chapter addresses design of the experiment, the experimental results, and the analysis of these results. The experiment had two factors and two levels for a total of four treatments. The resulting data set was analyzed using two-factor ANOVA. The data showed that the proactive tactic was superior to the reactive tactic. Riot leadership did not prove statistically to be a significant factor.

A.   DESIGN OF EXPERIMENT

The two factor two level experiment was designed with intent of analysis via two factor analysis of variance (ANOVA) with multiple observations per treatment or factor-level combination. The number of observations at each treatment was set at forty. The experimental model follows.

We draw on Devore (1994) for the essentials of two-factor ANOVA. He defines the following terms. Let I be the number of levels of the first factor or factor A, J the number of levels of factor B, and K the number of observations at each treatment. Then, let X_{ijk} be the kth observation at level i of the first factor, and level j of the second factor. Then,

$$X_{ijk} = \mu + \alpha_i + \beta_j + \gamma_{ij} + \varepsilon_{ijk}, \, i = 1, 2, j = 1, 2, k = 1, ..., 40.$$

The expected response, the grand mean over all levels of all factors, is μ. α_i is the effect due to the A at level i, β_j is the effect of B at level j, γ_{ij} is the interaction parameter, and ε_{ijk} is the error term for the kth observation at treament ij. We assume that the error terms are independent and normally distributed with variance σ^2. If the effects are additive, then all the γ_{ij}'s are 0. There are three sets of hypothesis that arise from this model:

1) H_{oAB}: $\gamma_{ij} = 0$ for all i, j   versus H_{aAB}: at least one $\gamma_{ij} \neq 0$

2) H_{oA}: $\alpha_1 = ... = \alpha_I = 0$ versus H_{oA}: at least one $\alpha_i \neq 0$

3) H_{oB}: $\beta_1 = ... = \beta_I = 0$ versus H_{oA}: at least one $\beta_i \neq 0$

The hypothesis of importance here is the null hypothesis H_{oA}. Specifically, the null hypothesis states there is no difference between the tactics employed. We desire to test to confidence level for $\alpha = 0.05$. The interaction parameter and the significance of leadership are also of interest and may yield further understanding of the process we are modeling.

B.   OUTPUT ANALYSIS

Let us review our two MOEs: MOE 1 is the expected number of hits taken by a peacekeeper; MOE 2 is the expected number hits taken by a rioter. The results of the two MOE's over the four scenarios are tabulated below. We see clearly that there are obvious differences between the two tactics.

	MOE 1 - Avg. Friendly Hits					MOE 2 - Avg. Enemy Hits		
		Leader					Leader	
		No	Yes				No	Yes
Tactic	React	2.09	1.48		Tactic	React	0.90	0.97
	Proactive	0.56	0.66			Proactive	0.45	0.54

Table 2. MOE Results

While the means are useful and provide observational insights, it is important to approach these results with an understanding of their statistical significance. The previously described ANOVA model was applied to the output data. S-PLUS 2000 provided the ANOVA reports shown in Appendix 2. The hypotheses and their p-values are summarized below.

Hypothesis	F-statistic		p-value		Reject Ho?	
	MOE 1	MOE 2	MOE 1	MOE 2	MOE 1	MOE 2
H_{oAB}	3.92	0.09	0.05	0.76	*yes*	*no*
H_{oA}	42.16	88.16	0.00	0.00	*yes*	*yes*
H_{oB}	2.03	2.82	0.16	0.10	*no*	*no*

Table 3. ANOVA Results Summary

The results showed that for both MOE's, the possibility that the two tactics had the same effect was essentially zero. We can safely reject the null hypothesis H_{oA}. The other hypotheses are a mixed bag. While the graphs below show that there are some conjectures we can make about the effect of crowd leadership, we cannot safely reject the hypothesis that leadership alone has no effect for either MOE. For MOE 1 the leadership did have an effect but only as an interaction with tactics.

Graphing the means leads to some interesting observations. In three of the four scenarios leadership resulted in having a higher level of confrontation, as measured by the number of hits on both sides. While the differences were not statistically significant, the trend appears compelling. What is surprising is that the reactive tactic had a better result against a led crowd than against a crowd lacking leadership. In retrospect this may result from the led crowd having only six stone throwers versus seven for the un-led crowd. The reduction in fighters could also explain the increased number of hits per rioter when the leader is present. The leader never gets fired upon with a reactive tactic since it never throws stones. And, since the reactive tactic does not allow firing unless fired upon, from the perspective of peacekeepers employing reactive tactics, leadership in the crowd ultimately meant one less stone thrower.
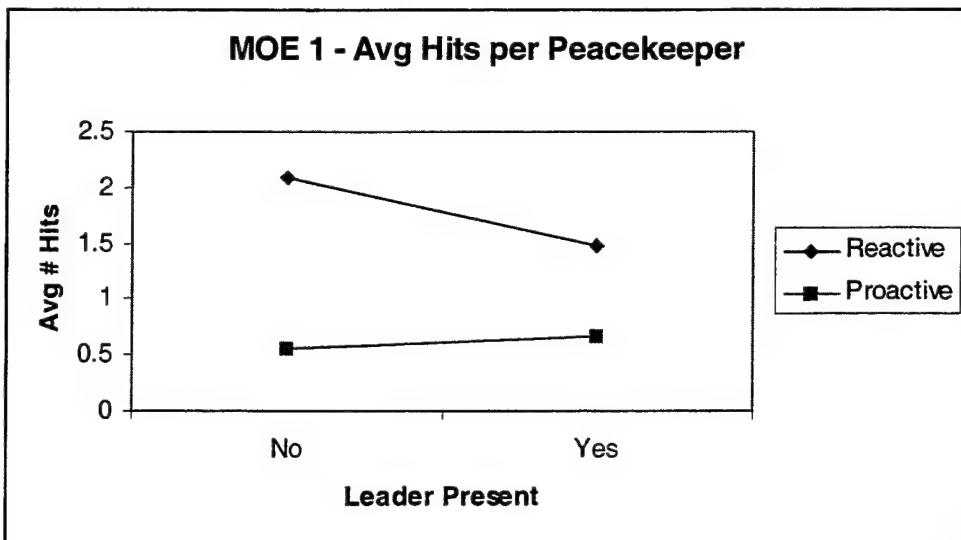
Figure 5. Average Hits per peacekeeper as a function of the presence of crowd leadership and peacekeeper tactic. Note the unexpected result of reactive tactics performing better against led crowds than those without.
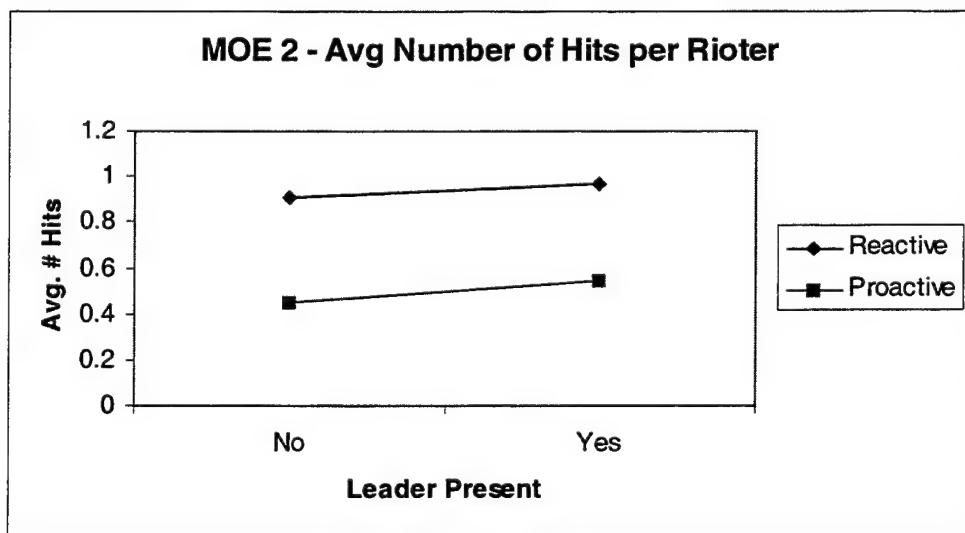
Figure 6. Average Number of Hits per Rioter. Note that the presence of the riot leader resulted in slightly more casualties for the rioters. This difference though proved not to be statistically significant.

Trading a riot leader for a rioter instead of augmenting the crowd with a leader was a decision made early on because the effect that was of concern was heat generation. Keeping the rioter and the leader would result in an obvious "heat" advantage to that level. Surrendering a stone thrower for a leader seemed to be a way of measuring the worth of the leadership. Additionally, making the leader a non-participant forced a

dichotomy between tactics that counter linear threats, the commensurate response, and tactics that seek to unravel the threat.

The result of this decision, coupled with small representative crowds due to run time limitations, was that adding a leader meant surrendering 14% of the firepower of the rioters. Ultimately, there is no reason to believe in the real world that riot leaders will never be active participants themselves. Also, even if they are non-participants, the numbers in large crowds are such to make the individual firepower contribution of a leader insignificant.

This skewing effect probably contributed to the fact that no effect for leadership was observed. The small number of entities involved may have also had a part in this. These smaller numbers probably contributed to greater variability per run, causing what appears to be a significant upward trend in casualties due to leadership (ignoring the skewed case) to be statistically insignificant.

To test this idea, a t-test was conducted by pooling the MOE 2 results into two groups: leadership and no leadership. The results are tabulated below.

Comparing # hits in MOE 2	No Leader	Leader
Mean	0.68	0.76
Variance	0.13	0.14
Observations	80	80
Pooled Variance	0.14	
Hypothesized Mean Difference	0	
Df	158	
t Stat	-1.35	
P(T<=t) one-tail	**0.09**	
t Critical one-tail	1.65	
P(T<=t) two-tail	0.18	
t Critical two-tail	1.97	

Table 4. Results of t-Test to examine if there still was a significant effect from leadership using only the MOE 2 results. Though provocatively low, the p-value does not allow one to reject the null hypothesis of no differences between the two cases.

While, it was hoped to be able to show a difference statistically, it was not the case. Perhaps this simulation model as currently configured fails to adequately represent the effect of crowd leadership. In light of the previous arguments, one can reasonably speculate that the leadership effect in this model is real and that given further experimentation the effect could be better demonstrated. If this is true, then the interaction term may actually be zero and only presented itself due to the skewed results from the lower firepower of the led crowd facing the reactive tactic.

The situation is reversed for proactive tactic. The proactive tactic attempts to anticipate violent action. Since the riot leader acts as a catalyst to the state transition, and the tactic employs a time step scanning process, the riot leader improves the crowd's ability to reach the transition point in large numbers prior to intervention by the peacekeepers. We can argue that it was this effect that led to an increase in violence for both MOE's when the riot leader was present, even though the means were not statistically different.

From operational experience, we know that leadership among the rioters will present a significant challenge to the peacekeepers (Allen, 1995). While the simulation failed to show that in this specific scenario, it did show that proactive tactics aimed at interrupting the process of riot formation were very successful at both protecting the peacekeepers and reducing the casualties of the rioters.

V. CONCLUSIONS AND RECOMMENDATIONS

This thesis developed and implemented an agent-based MOOTW small-scale combat modeling methodology. Using this implementation, called AgentKit, this thesis generated an increased understanding of the riot control problem. Using a few simple rules and the analogy of heat attracted particles, we generated a model of a riot control problem. By serving as a vehicle for experimentation, the model demonstrated the importance of violence supremacy in MOOTW encounters.

Agent-based simulation is a valuable method for the modeling of MOOTW small-scale combat. Using this methodology, researchers can generate complex behaviors from simple, reactive entities. Researchers can use these simulations to test ideas of how entities function in complex systems and evolve useful models describing their behavior. The simplicity of these individual models and the rules that govern their interactions grant transparency to the model. Transparency is an important attribute in agent-based simulation as it facilitates both the analysis of the model and the communication of its results. Agent-based models are best used as vehicles to improve intuition rather than as means to generate predictions.

In large-scale scenarios, the actions of individuals can be aggregated, as is done in most current warfare simulation. In MOOTW, individual actions have much greater potential for far-reaching impact. Agent-based models show the greatest potential for modeling the complex, small-scale scenarios of MOOTW where the actions of individuals are important. In addition to the analysis done above, the model developed here can be used to address other questions: what is the effect of adding another rioter to the riot leader scenario; do the results change when we vary the scale (20 versus 70); if force ratio is a predictor of outcome independent of scale, at what force ratio per tactic do the rioters always win; given such a force ratio, can we devise new tactics that will improve the situation for the peacekeepers?

The benefits of this kind of simulation are not so much in the answers they yield, but in the questions that their study generates. The results may seem obvious to military leaders with MOOTW operational experience. However, while the quantitative results

seem to validate some known aspects of riot control, the intent of the simulation was not to prove that proactive tactics are better than reactive tactics. The true intent of the simulation was to increase insight into this type of MOOTW scenario. By modeling at the entity level, the researcher is forced to study the scenario from the bottom up.

There are pitfalls in the use of agent-based simulation models. There is a temptation to perpetually increase the fidelity of the simulation, increasing the complexity of the agents, and the richness and diversity of the rules. This can lead to two different kinds of problem areas: transparency and extrapolation.

Transparency in agent-based simulation relates to the simplicity of the individuals, their rules, and the physics of their simulated world. While the aggregate behavior of these kinds of models may be complex, transparency of the individuals and their relationships contributes to understanding the roots of the complex behavior. As the modeler makes the individual agents more complex, the understanding of the cause and effect relationships in the model decreases.

Perhaps the most difficult part of developing an agent-based simulation model is finding the simple set of rules that make a useful first-order approximation of human behavior. It will always seem easier to add more rules, more features, and more functionality to the model in order to try and generate more realistic behavior. This will lead to more complexity but not necessarily more realism. A careful balance must be struck between the complexity necessary to capture a behavior and the simplicity necessary to understand what is happening in the model. Transparency has the additional benefit of aiding the researcher in conveying results to other parties. Ease of analysis for the researcher translates into ease of understanding for the end user.

The second problem is extrapolating the model results beyond its assumptions. Simulations of people can yield such seemingly realistic results, particularly if supplemented with animation, that users may be lulled into the belief that model is a faithful replica of reality. It must be kept in mind that merely because an agent-based model mimics a human systems behavior in a certain setting does not mean that in different settings the response will be valid. Use of these models for predictions must be done with great caution. Again, to paraphrase Axelrod (1997), the role of agent-based

simulations is not to predict the behavior of the system being modeled but to aid in its understanding.

Perhaps the greatest use for agent-based simulation will be in its synergistic application with other modeling methods. Analytical combat models, such as Many-on-Many stochastic duels using Semi-Markov chains, have as yet unexplored potential in modeling crowd control problems. Combining one or more analytical models of a MOOTW scenario with an agent-based simulation could yield far more insight into the process than either method alone.

In all likelihood, MOOTW will continue to be the environment requiring the most frequent application of U.S. military forces. It is unlikely to get less challenging. Agent-based modeling provides an important tool for combat modeling and experimentation in the MOOTW small-scale combat milieu. It is also the only tool currently available. Agent-based simulation deserves continued study and exploration as a valuable method for the modeling of MOOTW small-scale combat.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX. SOFTWARE USED IN THIS THESIS

All original software used in this simulation was written by the author in Java. Data generated by the simulation was analyzed using both Excel 2000 and S-PLUS 2000. The code for the simulation is contained in a package called AgentKit. As mentioned in the body of the thesis, AgentKit is an extension of Simkit (Stork, 1996). The code for the simulation's animation was a modification of Prof. Arnold H. Buss's animation package. This code is found in the package animationTest. The classes in these packages are listed below, AgentKit first, followed by animationTest. All the source code can be obtained by contacting Prof. Buss via his web page at **http://diana.or.nps.navy.mil/~ahbuss/**.

Directory of H:\AgentKit

07/31/00	11:13a	7,732 AgentReferee.java
07/19/00	02:50p	1,401 AgentRefereeManager.java
09/06/00	03:49p	2,326 AgentSimData.java
08/19/00	03:17p	8,826 BasicCombatAgent.java
08/21/00	12:41p	4,759 BasicCombatAgentMover.java
08/21/00	04:16p	2,468 BasicCreditHandler.java
08/15/00	05:38p	2,077 BasicSensorMessageGenerator.java
08/12/00	09:50a	2,451 BasicShooter.java
08/21/00	12:45p	3,913 BasicVisualSensor.java
07/28/00	03:35p	2,947 Chromosome.java
09/05/00	11:48p	921 CombatAgentContact.java
08/21/00	10:19a	11,443 CombatAgentMediator.java
08/21/00	04:16p	113 CreditHandler.java
08/02/00	03:28p	2,051 CreditRule.java
08/07/00	04:13p	1,460 DamageAssessor.java
08/21/00	12:37p	543 Detector.java
08/21/00	12:37p	571 Effector.java
08/16/00	11:22p	2,087 FireArm.java
07/25/00	09:36a	4,466 Gene.java
08/18/00	04:58p	8,988 HeatRadiator.java
08/13/00	10:13p	7,957 HeatRule.java
08/08/00	11:26p	56 LethalWeapon.java
08/16/00	11:21p	4,039 Message.java
03/29/00	02:50p	1,233 MessageEvent.java
08/23/00	03:15p	4,801 ModularMoverManager.java
07/23/00	10:20a	2,296 ModularShooterManager.java
08/21/00	10:21a	4,018 ModularWeaponManager.java

```
07/25/00  02:46p              307 Mortal.java
08/15/00  05:26p            2,125 MoverRule.java
08/15/00  05:27p           11,027 MoverRuleDefend.java
07/31/00  09:45a              546 MoverRuleDoNothing.java
08/08/00  11:44p            3,870 MoverRuleFlee.java
08/21/00  01:03p            3,194 MoverRuleFleeFiring.java
08/15/00  05:27p           11,885 MoverRuleFollow.java
08/21/00  12:04p           12,947 MoverRuleHeatAttracted.java
08/08/00  11:49p            5,360 MoverRulePath.java
08/08/00  11:50p            6,987 MoverRulePursuit.java
08/18/00  04:18p            3,985 MoverRuleRandom.java
08/02/00  11:48p            2,056 NonLethalFireArm.java
07/28/00  03:26p               59 NonLethalWeapon.java
08/21/00  12:37p            5,901 PerformanceSystem.java
08/21/00  09:22a              648 RiotLeaderHeatRadiator.java
08/09/00  10:00p              357 RiotLeaderHeatRule.java
07/28/00  02:24p               54 RiotWeapon.java
08/18/00  03:59p              114 Role.java
07/23/00  02:37p              944 Rule.java
05/02/00  03:56p              224 SensorMessageGenerator.java
07/31/00  12:31p              434 Shooter.java
08/10/00  03:16p            3,705 ShooterRule.java
07/31/00  09:44a              557 ShooterRuleDoNothing.java
08/21/00  10:20a            6,661 ShooterRuleNonLethal.java
09/06/00  03:50p            7,327 ShooterRuleNonLethalImproved.java
08/08/00  11:28p            5,894 ShooterRuleRandom.java
08/07/00  03:36p            7,418 ShooterRuleRioter.java
08/11/00  04:10p            3,984 ShooterRuleRioterRandom.java
08/12/00  12:04a            8,737 ShooterRuleROE.java
08/02/00  11:54p            2,073 StoneThrower.java
08/18/00  03:59p              612 TerrainObject.java
08/21/00  03:12p              781 UniformGenerator.java
07/23/00  11:04p              871 Weapon.java
              60 File(s)      217,587 bytes
```

Directory of H:\animationTest

```
09/12/00  10:02a           13,099 AgentAnimationTest.java
09/12/00  11:03a           11,441 AgentAnimationTest2.java
09/12/00  10:16a           11,370 animationDemo.java
08/16/00  11:04p            4,431 AnimationTest.java
08/16/00  04:03p            2,203 PingPanel.java
08/18/00  04:50p            3,862 PingThread2.java
08/16/00  04:08p            2,044 RandomLocationMoverManager.java
               7 File(s)       48,450 bytes
```

LIST OF REFERENCES

Joint Doctrine for Military Operations Other Than War, Joint Publication 3-07, 16 June 1995.

Appleget, J., "The Combat Simulation of Desert Storm with Applications for Contingency Operations", in *Warfare Modeling*, Bracken, J., Kress, M., Rosenthal, R. (Eds.) John Wiley & Sons, 1995.

Allen, J., "Humanity on Humanitarian Operations: How Much Violence is Enough?" *Marine Corps Gazette*, vol. 79, no. 2, pp. 14-21, Quantico, VA, 1995.

Andrade, S., "An Intelligent Agent Simulation of Shipboard Damage Control." Master's Thesis, Operations Research Department, Naval Postgraduate School, 2000.

Axelrod, R., *The Evolution of Cooperation*, BasicBooks, 1984.

Axelrod, R., *The Complexity of Cooperation*, Princeton University Press, 1997.

Clausewitz, Karl Von, *On War*, Trans. O.J. Matthijs Jolles, Random House, 1943.

Devore, J.L., *Probability and Statistics for Engineering and the Sciences*, 4th ed., Brooks/Cole Publishing Company, Pacific Grove, CA, 1995.

Ferber, J., *Multi-Agent Systems, an Introduction to Distributed Artificial Intelligence*, Addison-Wesley, 1999.

Fogel, D., *Evolutionary Computation, Toward a New Philosophy of Machine Intelligence*, IEEE, Inc., 2000.

Holland, J., *Hidden Order: How Adaptation Builds Complexity*, Helix Books, 1995.

Holland, J., *Emergence: From Chaos to Order*, Perseus Books, 1998.

Ilyachinski, A., *Land Warfare and Complexity, Part I: Mathematical Background and Technical Sourcebook (U)*, CNA Information Memorandum 96-461, July 1996.

Ilachinski, A., *Irreducible Semi-Autonomous Adaptive Combat (ISAAC): An Artificial-Life Approach to Land Combat*, CNA Research Memorandum 97-61.10, August 1997.

Kenny, J., "What's So Special About Non-Lethal Weapons' Human Effects - Everything," *Marine Corps Gazette*, vol. 84, no. 6, pp. 28-29, Quantico, VA, 2000.

Scharfen, J., "Dateline: Bosnia and Herzegovina September-November 1997," *Marine Corps Gazette*, vol. 82, no. 4, pp. 37-39, Quantico, VA, 1998.

Sears, F., Zemansky, M., and Young., H., *University Physics*, 5th ed., Addison-Wesley Publishing Company, 1976.

Stork, K., "Sensors in Object Oriented Discrete Event Simulation." Master's Thesis, Operations Research Department, Naval Postgraduate School, 1996.

Wagner, D., Mylander, W., Sanders, T. (Eds.), *Naval Operations Analysis*, 3rd Ed., Naval Institute Press, 1999.

Walter, D., "Warfighters Fight, Humanitarian Assist - Time to Match the Right Force to Humanitarian Assistance," *Marine Corps Gazette*, vol. 82, no. 4, pp. 40-42, Quantico, VA, 1998.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center...(2)
    8725 John J. Kingman Rd., STE 0944
    Ft Belvoir, VA 22060-6218

2. Dudley Knox Library...(2)
    Naval Postgraduate School
    411 Dyer Rd
    Monterey, CA 93943-5101

3. Director, Training and Education...(1)
    MCCDC, Code C46
    1019 Elliot Road
    Quantico, VA 22134-5027

4. Director, Marine Corps Research Center...(2)
    MCCDC, Code C40RC
    2040 Broadway Street
    Quantico, VA 22134-5107

5. Director, Studies and Analysis...(1)
    MCCDC Code C45
    300 Russell Road
    Quantico, VA 22134-5130

6. Marine Corps Representative...(1)
    Naval Postgraduate School
    Code 037, Bldg. 330, Ingersoll Hall, Room 116
    555 Dyer Road
    Monterey, CA 93943

7. Marine Corps Tactical Systems Support Activity...(1)
    Technical Advisory Branch
    Attn: Librarian
    Box 555171
    Camp Pendleton, CA 92055-5080

8. Professor Arnold H. Buss ....(3)
    Code OR/Bu
    Operations Research Department
    U.S. Naval Postgraduate School
    Monterey, CA 93943

9. Professor Gordon H. Bradley...(1)
Code OR/Br
Operations Research Department
U.S. Naval Postgraduate School
Monterey, CA 93943

10. Lecturer John Hiles...(1)
Code CS/Hi
Computer Science Department
U.S. Naval Postgraduate School
Monterey, CA 93943

11. Professor Paul J. Sanchez..(1)
Code OR/Sp
Operations Research Department
U.S. Naval Postgraduate School
Monterey, CA 93943

12. Doctor Moshe Kress..(1)
CEMA (T1)
P.O. Box 2250
Haifa 31021
Israel

13. Major Ronald F. A. Woodaman, USMC..(3)
13706 Shelburne St.
Centreville, VA 90120